```
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git add src/main/java/com/qa/ims/IMS.java
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git commit -m "JA: Added all item features to item features branch and com
mited"
[item_features 4c77a06] JA: Added all item features to item features branch and commited
 4 files changed, 305 insertions(+)
 create mode 100644 src/main/java/com/qa/ims/controller/ItemController.java
 create mode 100644 src/main/java/com/qa/ims/persistence/dao/ItemDAO.java
 create mode 100644 src/main/java/com/qa/ims/persistence/domain/Item.java
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git push origin item_features
```

Commit and push

```
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git checkout development
M       src/main/java/com/qa/ims/Runner.java
M       src/main/resources/db.properties
Switched to branch 'development'
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git merge item_features
Updating c27d792..4c77a06
Fast-forward
 src/main/java/com/qa/ims/IMS.java                          |   6 +++
 src/main/java/com/qa/ims/controller/ItemController.java    |  70 ++++++++++++++++++++++++++++++++++
 src/main/java/com/qa/ims/persistence/dao/ItemDAO.java      | 126 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 src/main/java/com/qa/ims/persistence/domain/Item.java      | 103 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 305 insertions(+)
 create mode 100644 src/main/java/com/qa/ims/controller/ItemController.java
 create mode 100644 src/main/java/com/qa/ims/persistence/dao/ItemDAO.java
 create mode 100644 src/main/java/com/qa/ims/persistence/domain/Item.java

Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git push origin development
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JehadAbdelBaqi/IMS_PROJECT.git
   c27d792..4c77a06  development -> development
```

Add, commit and push merge into development.

```
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git branch
* development
  item_features
  main
  order_features
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$ git checkout order_features
M       src/main/java/com/qa/ims/Runner.java
M       src/main/resources/db.properties
Switched to branch 'order_features'
Jehads-MacBook-Pro:IMS_PROJECT jehadabdelbaqi$
```


ture branch and repeat the process.

Testing

Application testing

Application testing using Junit and Mockito.

ItemDAOTest.java | Item.java | sql-schema.sql | sql-data.sql

```java
package com.qa.ims.persistence.dao;

import static org.junit.Assert.assertEquals;

public class ItemDAOTest {

    private final ItemDAO DAO = new ItemDAO();

    @Before
    public void setup() {
        DBUtils.connect();
        DBUtils.getInstance().init("src/test/resources/sql-schema.sql", "src/test/resources/sql-data.sql");
    }

    @Test
    public void testCreate() {
        final Item created = new Item(2L, "XBOX", 2000.0);
        assertEquals(created, DAO.create(created));
    }

    @Test
    public void testReadAll() {
        List<Item> expected = new ArrayList<>();
        expected.add(new Item(1L, "Laptop", 999.99));
        assertEquals(expected, DAO.readAll());
    }

    @Test
    public void testReadLatest() {
        assertEquals(new Item(1L, "Laptop", 999.99), DAO.readLatest());
    }

    @Test
    public void testRead() {
        final long ID = 1L;
        assertEquals(new Item(ID, "Laptop", 999.99), DAO.read(ID));
    }

    @Test
    public void testUpdate() {
        final Item updated = new Item(1L, "XBOX", 2000.0);
        assertEquals(updated, DAO.update(updated));
    }

    @Test
    public void testDelete() {
        assertEquals(1, DAO.delete(1));
    }
}

//java.lang.AssertionError: expected:<[Item [id=1, itemName=Laptop, itemPrice=999.99]]>
//but was:<[Item [id=1, itemName=Laptop, itemPrice=999.99], Item
//          [id=2, itemName=Laptop, itemPrice=999.99], Item
//          [id=3, itemName=Laptop, itemPrice=999.99]]>
```

CodeTogether...

ItemControllerTest.java

```
30      private ItemController controller;
31
32      @Test
33      public void testCreate() {
34          final String I_NAME = "XBOX";
35          final Double I_PRICE = 1000.0;
36          final Item created = new Item(I_NAME, I_PRICE);
37
38          Mockito.when(utils.getString()).thenReturn(I_NAME);
39          Mockito.when(utils.getDouble()).thenReturn(I_PRICE);
40          Mockito.when(dao.create(created)).thenReturn(created);
41
42          assertEquals(created, controller.create());
43
44          Mockito.verify(utils, Mockito.times(1)).getString();
45          Mockito.verify(utils, Mockito.times(1)).getDouble();
46          Mockito.verify(dao, Mockito.times(1)).create(created);
47      }
48
49      @Test
50      public void testReadAll() {
51          List<Item> items = new ArrayList<>();
52          items.add(new Item(1L, "PS5", 2000.0));
53
54          Mockito.when(dao.readAll()).thenReturn(items);
55
56          assertEquals(items, controller.readAll());
57
58          Mockito.verify(dao, Mockito.times(1)).readAll();
59      }
60
61      @Test
62      public void testUpdate() {
63          Item updated = new Item(1L, "Laptop", 3000.0);
64
65          Mockito.when(this.utils.getLong()).thenReturn(1L);
66          Mockito.when(this.utils.getString()).thenReturn(updated.getItemName());
67          Mockito.when(this.utils.getDouble()).thenReturn(updated.getItemPrice());
68          Mockito.when(this.dao.update(updated)).thenReturn(updated);
69
70
71          assertEquals(updated, this.controller.update());
72
73          Mockito.verify(this.utils, Mockito.times(1)).getLong();
74          Mockito.verify(this.utils, Mockito.times(1)).getString();
75          Mockito.verify(utils, Mockito.times(1)).getDouble();
76          Mockito.verify(this.dao, Mockito.times(1)).update(updated);
77      }
78
79      @Test
80      public void testDelete() {
81          final long ID = 1L;
82
83          Mockito.when(utils.getLong()).thenReturn(ID);
84          Mockito.when(dao.delete(ID)).thenReturn(1);
85
86          assertEquals(1L, this.controller.delete());
87
```

Demo and User story discussoion

Demonstration

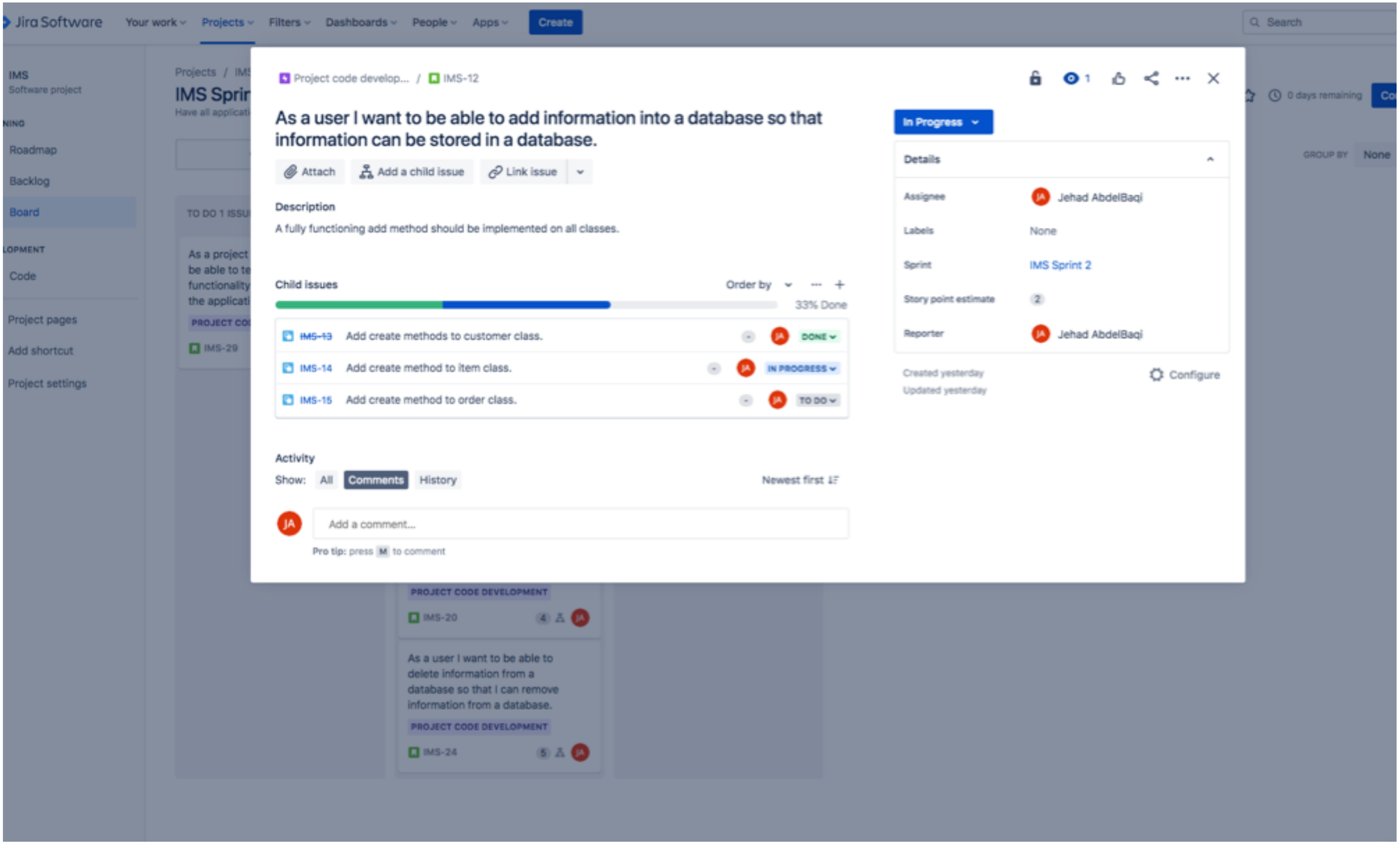
Demonstrate the application and discuss some user stories and software implementation.

ItemDAO.java | ItemController. | CustomerDAO.jav

```
27      }
28
29      /**
30       * Reads all customers from the database
31       *
32       * @return A list of customers
33       */
34      @Override
35      public List<Customer> readAll() {
36          try (Connection connection = DBUtils.getInstance(
37                  Statement statement = connection.createSt
38                  ResultSet resultSet = statement.executeQu
39              List<Customer> customers = new ArrayList<>();
40              while (resultSet.next()) {
41                  customers.add(modelFromResultSet(resultSe
42              }
43              return customers;
44          } catch (SQLException e) {
45              LOGGER.debug(e);
46              LOGGER.error(e.getMessage());
47          }
48          return new ArrayList<>();
49      }
50
51      public Customer readLatest() {
52          try (Connection connection = DBUtils.getInstance(
53                  Statement statement = connection.createSt
54                  ResultSet resultSet = statement.executeQu
55              resultSet.next();
56              return modelFromResultSet(resultSet);
57          } catch (Exception e) {
58              LOGGER.debug(e);
59              LOGGER.error(e.getMessage());
60          }
61          return null;
```


One example here is the customer DAO which has specific functionality that makes a connection to the database and can either create, read, update or delete any information in a database.

This was also implemented into the item and order table classes. These methods were made based upon the user stories created in the project management boards.



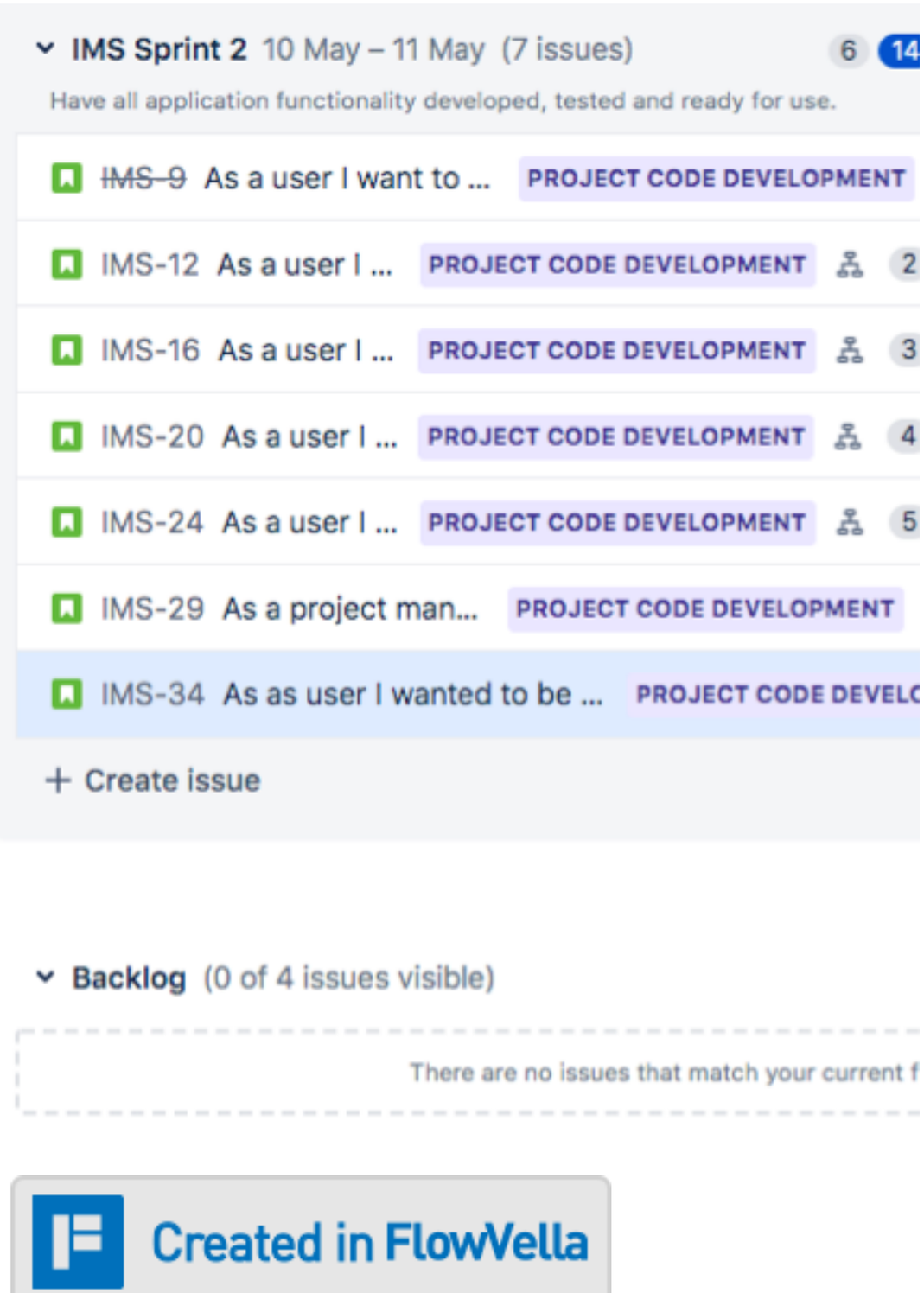
The user stories amongst others were used to create the necessary functionalities for the program.

Sprint review and retrospective

Spring

What we managed to complete, what was left behind and what could have been done better.

Sprint review



Upon revision of the project passed I felt many things could have been done better. With the limited time given to work on the project some things have been left out.

I decided to stick to CRUD functionality to ensure the program runs as it's supposed to.

Testing wasn't completed but application was running with all methods working.

Some things will definitely need to be improved with some methods refactored.

Retrospective

The project went well - although many improvements could be made.

M The application MUST have what was outlined in the initial specification such as database CRUD functionality and all relevant documentation.

O The project SHOULD have more database entries within the tables such as more information regarding the customers and the items. IE contact details for customers and quantity of items,

C The project COULD have user limitations depending on who is using the application. For example a customer should only be able to access his own information while admin have more access and editing permissions.

O

W The project WONT have the user limitations due to time constraints which means the current application can only be used by administrative users who can make changes to all entries.